Lab

Arrays in C

- An array is defined as the collection of similar type of data items stored at contiguous memory locations.
- Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- It also has the capability to store the collection of derived data types, such as pointers, structure, etc.
- The array is the simplest data structure where each data element can be randomly accessed by using its index number.

Properties of Array:

1. Each element of an array is of same data type and carries the same size.
2. Array element starting index is 0 and last index is **size of array – 1.**
3. Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
4. Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

Disadvantage of C Array:

1. Fixed Size: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList.

Declaration of C Array:

data_type array_name[array_size];

Example:

int marks[5];

where marks is an array name and data type of marks is int and size of array marks is 5 i.e.

marks can store max 5 int elements [total 10 bytes].

Initialization of C Array:

Array elements can be initialized by i) compile time initialization ii) run time initialization

Compile time initialization:

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

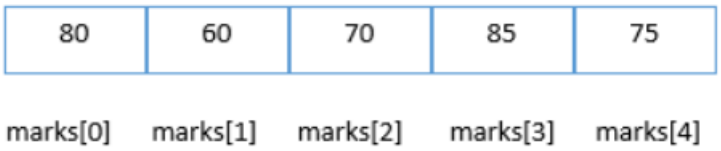
marks[0]=80;//initialization of array

marks[1]=60;

marks[2]=70;

marks[3]=85;

marks[4]=75;



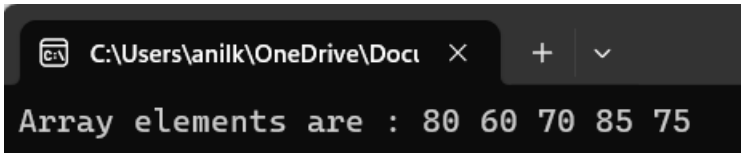
Initialization of Array

//C Program to initialize array elements – compile time initialization

```
#include<stdio.h>
int main()
{
	int i=0;
	int marks[5];//declaration of array
	marks[0]=80;//initialization of array
	marks[1]=60;
	marks[2]=70;
	marks[3]=85;
	marks[4]=75;

	//traversal of array
	printf("Array elements are : ");
	for(i=0;i<5;i++)
	{
		printf("%d ",marks[i]);	// display each element in the array
	}
	return 0;
}
```

Output:



Alternate way of compile time initialization:

int marks[5]={20,30,40,50,60}; //C array declaration with initialization

Runtime initialization:

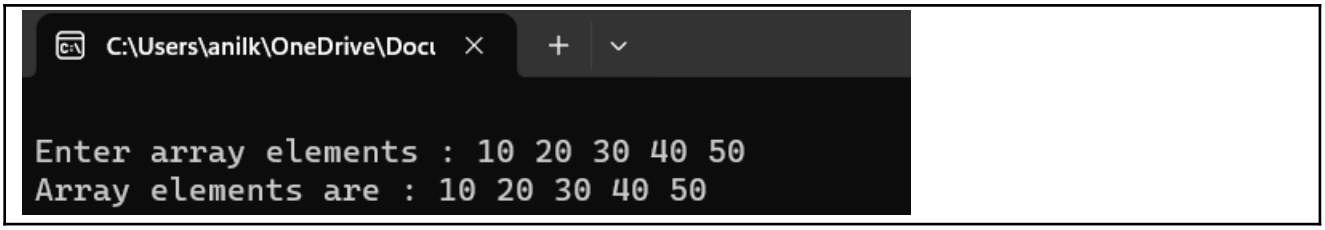
Using scanf() function we can initialize array elements at run time(during program execution).

// program to initialize array elements at runtime

```
#include<stdio.h>
int main()
{
        int i=0;
        int marks[5];//declaration of array
         //reading array elements
        printf("\nEnter array elements : ");
       for(i=0;i<5;i++)
       {
       scanf("%d",&marks[i]);
       }

       //printing array elements
      printf("Array elements are : ");
      for(i=0;i<5;i++)
      {
            printf("%d ",marks[i]);    // display each element in the array
      }
      return 0;
}
```

Output:



Single Dimensional Arrays:

UNIT III

WEEK 7:

Objective: Explore the full scope of Arrays construct namely defining and initializing 1-D and 2-D and more generically n-D arrays and referencing individual array elements from the defined array. Using integer 1-D arrays, explore search solution linear search.

Suggested Experiments/Activities:

Tutorial 7: 1 D Arrays: searching.

Lab 7:1D Array manipulation, linear search

i) Find the min and max of a 1-D integer array.

ii) Perform linear search on1D array.

iii) The reverse of a 1D integer array

iv) Find 2's complement of the given binary number.

v) Eliminate duplicate elements in an array.

Write an algorithm to flowchart to find the minimum and maximum elements in the given array

```
/* C program to find the minimum and maximum elements in the given list of elements */

#include<stdio.h>
int main()
{
        int a[100],min,max,i,size;
        printf("Enter the count of elements : ");
        scanf("%d",&size);

        printf("Enter the elements : ");
        for(i=0;i<size;i++)    // reading elements
            scanf("%d",&a[i]);
```

```
        printf("\nList of elements are : ");
        for(i=0;i<size;i++)    // printing elements
            printf("%d ",a[i]);

        min=a[0];
        for(i=0;i<size;i++)   // finding minimum element in the given array
        {
            if(min>a[i])
                  min=a[i];
        }
        printf("\nMinimum element in the given list is %d ",min);

        max=a[0];
        for(i=0;i<size;i++)   // finding minimum element in the given array
        {
            if(max<a[i])
                max=a[i];
        }
        printf("\nMaximum element in the given list is %d ",max);
        return 0;
}
```

Output:

```
Enter the count of elements : 5
Enter the elements : 10 20 15 45 75 55

List of elements are : 10 20 15 45 75
Minimum element in the given list is 10
Maximum element in the given list is 75
```

Write an algorithm and flowchart to search for an element using linear search

```
/* C program to search an element from the given list of elements using linear search */

#include<stdio.h>
int main()
{
        int a[100],i,size,key,flag=0;
        printf("Enter the count of elements : ");
        scanf("%d",&size);

        printf("Enter the elements : ");
        for(i=0;i<size;i++)    // reading elements
            scanf("%d",&a[i]);

        printf("List of elements are : ");
        for(i=0;i<size;i++)    // printing elements
            printf("%d ",a[i]);

        printf("\nEnter the element to be searched : ");   //search element
        scanf("%d",&key);

         for(i=0;i<size;i++)    // Linear Search
         {
             if(a[i]==key)    // key value is identified
             {
                 flag=1;
                 break;
             }
         }

          if(flag==0)
              printf("Element is not present in the given list");
          else
              printf("Element is present at location %d",i+1);
          return 0;
}
```

Output:

```
Enter the count of elements : 6
Enter the elements : 10 20 15 56 78 90
List of elements are : 10 20 15 56 78 90
Enter the element to be searched : 56
Element is present at location 4
```

Write an algorithm and flowchart to reverse the given 1D array

```
//  C program to reverse the given 1D array

#include<stdio.h>
int main()
{
        int a[100],b[100],i,size,temp;
        printf("Enter the array size : ");
        scanf("%d",&size);

        printf("Enter the array elements : ");
        for(i=0;i<size;i++)    // reading elements
           scanf("%d",&a[i]);

        printf("Original array is : ");
        for(i=0;i<size;i++)    // printing original array elements
           printf("%d ",a[i]);

        for(i=0;i<size/2;i++)    //reverse of arrray logic
        {
                temp=a[i];
                a[i]=a[size-i-1];
                a[size-i-1]=temp;
        }

        printf("\nReverse of the given array is : ");
        for(i=0;i<size;i++)    // reading elements
           printf("%d ",a[i]);
        return 0;
}
```

Output:

```
Enter the array size : 5
Enter the array elements : 10 20 30 40 50
Original array is : 10 20 30 40 50
Reverse of the given array is : 50 40 30 20 10
```

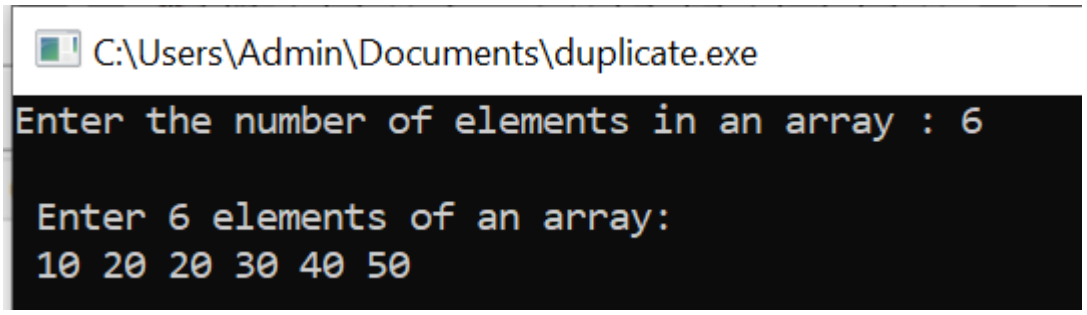
Write an algorithm and flowchart to remove the duplicate elements in the given array

```
#include<stdio.h>
int main ()
{
      int arr[20],i,j,k,size;
     printf("Enter the number of elements in an array : ");
     scanf("%d", &size);
    printf("\n Enter %d elements of an array: \n ", size);
    // reading elements into the array
    for(i=0;i<size;i++)
   {
     scanf("%d",&arr[i]);
   }
    // removing duplicate elements in the array
   for(i=0;i<size;i++)
   {
      for(j=i+1;j<size;j++)
      {
         if(arr[i]==arr[j])  // check duplicate
         {
            // delete the current position of duplicate element
            for(k=j;k<size-1;k++)
            {
               arr[k]=arr[k+1];
            }
    // decrease the size of array after removing duplicate element
            size--;


/* if the position of the elements is changes, don't increase the
index j */
            j--;
         }
      }
   }
```

```
   /* display an array after deletion or removing of the duplicate elements */
   printf (" \n Array elements after deletion of the duplicate elements: ");

   // for loop to print the array
   for ( i = 0; i < size; i++)
   {
      printf (" %d \t", arr[i]);
   }
   return 0;
}
```



WEEK 8:

Objective: Explore the difference between other arrays and character arrays that can be used as Strings by using null character and get comfortable with string by doing experiments that will reverse a string and concatenate two strings. Explore sorting solution bubble sort using integer arrays.

Suggested Experiments/Activities:

Tutorial 8: 2 D arrays, sorting and Strings.

Lab 8: Matrix problems, String operations, Bubble sort

i) Addition of two matrices

ii) Multiplication two matrices

iii) Sort array elements using bubble sort

iv) Concatenate two strings without built-in functions

v) Reverse a string using built-in and without built-in string functions

Basics:

Two Dimensional Arrays:

syntax:
int array_name[row_size][column_size];
int x[5][10]; // array x with 5 rows and 10 columns
int x[3][3]; // array x with 3 rows and 3 columns

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

The above matrix contains three rows and three columns

Row index starts from 0

Column index starts from 0

Initialization of 2D arrays:

i) Compile time initialization ii) Run time initialization

i) Compile time initialization:

int a[3][2]={10,20,30,40,50,60}; // total 6 elements

int a[3][2]={{10,20},{30,40},{50,60}}; //total 6 elements

Printing 2D array values:

printf(“%d”,a[0][0]); //print element of row0 and column0

printf(“%d”,a[0][1]); //print element of row0 and column1

printf(“%d”,a[1][0]); //print element of row1 and column0

printf(“%d”,a[1][1]); //print element of row1 and column1

printf(“%d”,a[2][0]); //print element of row2 and column0

printf(“%d”,a[2][1]); //print element of row2 and column1

ii) Run time initialization:

We can initialize the values into the 2D array at the time of execution.

int a[3][2]; // array a with 3 rows and 2 columns

Reading value from the keyboard using scanf() function:

printf(“Enter any element : “);

// reading values into row0 and column1

scanf(“%d”,&a[0][1]); // reading values into row0 and column2

scanf(“%d”,&a[1][0]); // reading values into row1 and column0

scanf(“%d”,&a[1][1]); // reading values into row1 and column1

scanf(“%d”,&a[2][0]); // reading values into row2 and column0

scanf(“%d”,&a[2][1]); // reading values into row2 and column0

Write a C program to read and write values into 2D array

```
// C program to read and write values into 2D array
/*C program to read and write values into the 2D array*/
#include<stdio.h>
int main()
{
        int a[50][50],m,n,i,j;
        printf("Enter the dimensions of a matrix: ");
        scanf("%d%d",&m,&n);
        //reading elements into the 2D array

        printf("\nEnter the elements \n");
        for(i=0;i<m;i++)    // m-rows
        {
            for(j=0;j<n;j++)  // n-columns
            {
                scanf("%d",&a[i][j]);
            }
        }

        printf("\Matrix Elements are : ");
        for(i=0;i<m;i++)    // m-rows
        {
             printf("\n");
            for(j=0;j<n;j++)  // n-columns
            {
                printf("%d  ",a[i][j]);
             }
    }
 }
```

Output:

```
D:\MNRAO\2d.exe
Enter the dimensions of a matrix: 3 4

Enter the elements
10 20 30 40
50 60 70 80
81 82 83 84
Matrix Elements are :
10  20  30  40
50  60  70  80
81  82  83  84
--------------------------------
```

Matrix Addition:

C=A+B

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1+9 & 2+8 & 3+7 \\ 4+6 & 5+5 & 6+4 \\ 7+3 & 8+2 & 9+1 \end{bmatrix}$$